Public Transp (2017) 9:73–94
DOI 10.1007/s12469-016-0143-x

ORIGINAL PAPER

A branch-and-bound approach for robust railway timetabling

Gábor Maróti[1]

Accepted: 2 November 2016 / Published online: 21 November 2016

Abstract This paper studies the robust cyclic timetabling problem. The goal is to modify a given reference timetable to enhance its robustness against small stochastic disturbances. The robustness is measured by the expected total delay of the realised timetable. Kroon et al. (Transp Res Part B 42(6):553–570, 2008) propose a stochastic programming approach and implement it for Netherlands Railways (NS). While the model's outcome is accepted by practitioners, relevant planning problems are rendered intractable by computation times of up to several days. In this paper we describe a Branch-and-Bound algorithm for solving the stochastic program of Kroon et al. (Transp Res Part B 42(6):553–570, 2008). We propose specific node selection rules, variable selection rules, constructive heuristics and lower bounds. We carry out computational tests on large real-life problem instances. The results confirm that our algorithm is able to considerably improve the robustness of the reference solutions. This is achieved with computation times of a few minutes. However, the weak lower bounds we use leave a considerable optimality gap. Therefore, our algorithm is best described as a heuristic solution method.

Keywords Robust timetabling · Stochastic programming · Branch-and-bound

1 Introduction

In this paper we study the robust cyclic timetabling problem. We aim at modifying a given *reference timetable* in such a way that the sum of the realised delays is minimised when operating it subject to stochastic travel times. Our main focus lies in designing a pragmatic approach to solving large, practically relevant cases.

✉ Gábor Maróti
g.maroti@vu.nl

[1] VU University Amsterdam and Netherlands Railways, Amsterdam, The Netherlands

Cyclic timetabling (also known as periodic timetabling) specifies the timetable for a longer period (e.g., a day) as repeated copies of the timetable for a shorter period (e.g., for an hour). This is a common practice in public transportation, such as at passenger rail operators and in urban mass transit networks.

Cyclic timetabling is an intensively studied problem. Serafini and Ukovich (1989) introduce the periodic event scheduling problem (PESP). Schrijver and Steenbeek (1994) describe a constraint propagation approach. Nachtigall and Opitz (2008) develop the modulo simplex algorithm for PESP. Liebchen (2006) designs a Mixed Integer Programming formulation of PESP to solve large real-life cases to optimality. These cases include the Berlin subway system and the 1-h timetable of Netherlands Railways (which is the underlying problem of this paper).

Recently, Kümmling et al. (2015) propose the use of SAT solvers for PESP problems. Further details about cyclic timetabling can be found in Caprara et al. (2007), Kroon et al. (2014), Peeters (2003) and Liebchen (2006).

The planning models for cyclic timetabling assume deterministic travel times between any pair of stations. In practice, however, the travel times are stochastic; the variation can be attributed to weather conditions, to fluctuating embarking and disembarking times, or to minor technical mishaps. The *travel time supplement*, also known as buffer time, of a timetable is to absorb as much of these disturbances as possible, the non-absorbable part is experienced as delay, and is propagated to later events.

We want to emphasise that this research focuses on the resilience of the timetable against small disturbances. The only way to fight delays is by using the time supplement. Large-scale disruptions, on the other hand, require substantial changes to the timetable, such as cancellation and re-routing of trains. Moreover, wait-or-no-wait decisions are to be made which is the topic of the intensively studied field of delay management. Large-scale disruptions lie beyond the scope of this paper.

In this paper we study the problem on the macro level; our approach is built around a macro-simulation. The rich field of micro-simulations is beyond our scope.

Cacchiani and Toth (2012) provide a survey of recent results of nominal and robust timetabling. Shafia et al. (2012) consider the robust timetabling problem of a single-track railway line, and describe exact as well as heuristic solution algorithms. Goerigk (2014) studies the robust PESP in the context of recoverable robustness, and proposes a heuristic algorithm.

Kroon et al. (2008) describe a stochastic programming model. The aim is to distribute the available time supplement between the train rides so that the expected total delay is minimised. The stochastic program is solved via a Mixed Integer Programming formulation. Computational results on real-life instances of Netherlands Railways (NS) prove the practical value of the model. NS has been using SOM, a commercial implementation of the model, since 2014.

As an alternative to stochastic programming, Fischetti et al. (2009) introduce the concept of light robustness and solve an optimisation model that distributes the available time supplement according to a heuristic objective. The solutions of the proposed model are competitive with the solutions of the stochastic programming approach, but require solution times that are orders of magnitude smaller.

Cicerone et al. (2009) study robust timetabling within the framework of recoverable robustness, with a primary focus on theoretical results. Liebchen et al. (2010) extend light robustness slightly to compute delay-resistant periodic timetables, the focus being on a delay management objective.

In this paper we consider the stochastic program proposed by Kroon et al. (2008) and we describe an alternative way for solving it. The solution method of Kroon et al. (2008) scales poorly as the problem size increases; practically relevant instances lead to computation times of several days. Our main goal is to design a solution method that can solve large real-life instances much faster, preferably within a few minutes. Such an improvement would enable practitioners to use the robust timetabling software tool much more frequently than it happens now.

We propose a Branch-and-Bound algorithm for solving the underlying stochastic integer program. The lower bounds of the Branch-and-Bound nodes arise from a variant of macro-simulation; we do not rely on any linear or non-linear programming relaxation. The generic Branch-and-Bound framework is fine-tuned by proposing specific node selection rule and variable selection rules as well as an auxiliary quadratic optimisation model for finding feasible solutions. We demonstrate the power of our methods on real-life instances of NS.

Our method is able to find solutions of good quality very quickly, in a matter of minutes. However, the weak lower bounds leave a significant optimality gap, often in the order of 20–30%. Therefore, our algorithm can best be described as a heuristic solution method.

Although our paper studies railway timetabling, the main ideas are likely to be applicable for a wider class of stochastic integer programming problems. We do admit, though, that the proposed algorithm is fine-tuned for the problem instances of NS. We expect that other application areas could require different node selection and variable selection rules.

The contributions of this paper are summarised as follows.

- We study the robust railway timetabling problem.
- We describe a Branch-and-Bound algorithm for the stochastic optimisation model.
- We propose application-specific rules in the Branch-and-Bound framework.
- We illustrate the usefulness of our method on very large problem instances of Netherlands Railways.

This paper is organised as follows. Section 2 provides a detailed problem description. We describe the solution method in Sect. 3. Section 4 is devoted to our computational results. Finally, we draw some conclusions and outline our future research in Sect. 5.

2 Problem description

In this section we describe the underlying robust cyclic timetabling problem. The setting is identical to the formulation in Kroon et al. (2008) as well as in its commercial implementation at NS. The only difference between the model of Kroon

et al. (2008) and that of this paper is a straightforward transformation of the decision variables.

Throughout the paper we assume that a *reference timetable* is given. Our aim is then to modify the reference timetable slightly so that its robustness is enhanced.

2.1 Reference timetable

Cyclic timetabling assumes a cycle time T, and aims at finding the timetable for a time period of length T. A day's timetable is composed of a number of repeated copies of the single-period timetable. For the sake of a simpler terminology, we assume that the cycle time is 1 h (i.e., $T = 60$ min) which is the case in the Dutch railway system.

We will use the following notational convention. Symbols for variables and parameters are decorated by a circle if they refer to the *cyclic* timetable. In this way, we create a visual distinction to the symbols related to the *linear* timetable (which is described in Sect. 2.3).

The fundamental objects of a cyclic timetabling problem are the *cyclic events*; they form the set $\mathscr{E}^\circ$. A cyclic event corresponds to the departure or the arrival of a given timetable service at a given station. The *reference timetable* ρ° itself is an assignment of the cyclic events to time instants (subject to restrictions discussed below).

The cyclic time scale means that the planned time of $\mathring{e} \in \mathscr{E}^\circ$ in timetable ρ° satisfies $0 \leqslant \rho^\circ(\mathring{e}) < T$. Driven by our application we assume that the planned event times are whole minutes in the interval $[0; T-1]$. That is, the reference timetable is a function $\rho^\circ : \mathscr{E}^\circ \to \{0, \ldots, T-1\}$.

The restrictions of the cyclic timetabling problem are expressed by the *cyclic processes*; they form the set $\mathscr{P}^\circ$. Cyclic processes can model the travel of a train between two neighbouring stations, the dwelling of a train, headway restrictions between different trains, transfer opportunities for passengers, and so on. For further details about the cyclic timetabling we refer to Vromans (2005).

A cyclic process imposes a lower bound and an upper bound on the time difference of two cyclic events. The cyclic time scale leads to the following constraints.

$$\ell^\circ(\mathring{p}) \leqslant \rho^\circ(\mathring{e}') - \rho^\circ(\mathring{e}) + Q(\mathring{p}) \cdot T \leqslant u^\circ(\mathring{p}) \qquad \text{for each } \mathring{p} = (\mathring{e}, \mathring{e}') \in \mathscr{P}^\circ \qquad (1)$$

The values Q are integer parameters to indicate the cyclic order of the events. We assume that the time duration of a cyclic process is always less than T, and that the lower and upper bounds satisfy the relations $0 \leqslant \ell^\circ(\mathring{p}) < T$ and $\ell^\circ(\mathring{p}) \leqslant u^\circ(\mathring{p}) < \ell^\circ(\mathring{p}) + T$. Under these assumptions, the value of $Q(\mathring{p})$ can be chosen from the set $\{0, 1, 2\}$.

The lower bounds ℓ° indicate the *technically minimal process times*. The difference between the planned process time and the technically minimal process time is the *process time supplement*. The upper bounds are used for ensuring a safe headway time between the trains, and for expressing marketing requirements such as making the travel times attractively low.

2.2 Modified timetable

The robust timetabling problem, as proposed by Kroon et al. (2008), aims at finding a new timetable by slightly modifying the reference timetable. A particularly important restriction is that the modified timetable keeps the cyclic order of the events the same as it is in the reference timetable. This restriction has a two-fold motivation. On one hand, the unrestricted cyclic timetabling problem barely tractable, even for a linear objective function. On the other hand, the robust timetabling model fits well to the planning work-flow: it is used to fine-tune a timetable once the crucial decisions on its basic shape have been taken.

The modifications of the reference timetable are expressed by the *event shift variables* $x : \mathscr{E}^\circ \to \mathbb{Z}$. Then the planned event time of $\mathring{e}$ becomes $\rho^\circ(\mathring{e}) + x(\mathring{e})$.

We consider the following cyclic timetabling problem.

$$-m(\mathring{e}) \leqslant x(\mathring{e}) \leqslant m(\mathring{e}) \quad \text{for each } \mathring{e} \in \mathscr{E}^\circ \tag{2}$$

$$\bar{\ell}^\circ(\mathring{p}) \leqslant \; x(\mathring{e}') - x(\mathring{e}) \leqslant \bar{u}^\circ(\mathring{p}) \quad \text{for each } \mathring{p} = (\mathring{e}, \mathring{e}') \in \mathscr{P}^\circ \tag{3}$$

$$x(\mathring{e}'_s) - x(\mathring{e}_s) \leqslant M_s \quad \text{for each } s \in \mathscr{S} \tag{4}$$

$$\sum_{s \in \mathscr{S}} \big(x(\mathring{e}'_s) - x(\mathring{e}_s)\big) \quad \leqslant M \tag{5}$$

Constraints (2) limit the deviation from the reference timetable. The values m are set to 1 min in our application. This choice is pragmatic: it leads to the best performance of our algorithm, both in terms of running time and in terms of solution quality. At the same time, $m = 1$ fits well to the concept of a slight timetable modification, and provides practically meaningful solutions.

Constraints (3) are the reformulation of (1), stating that the modified timetable $\rho^\circ + x$ must regard the lower and upper bounds on the planned process times. The bounds of $\mathring{p} = (\mathring{e}, \mathring{e}') \in \mathscr{P}^\circ$ are defined as

$$\bar{\ell}^\circ(\mathring{p}) = \ell^\circ(\mathring{p}) - \rho^\circ(\mathring{e}') + \rho^\circ(\mathring{e}) - Q(\mathring{p}) \cdot T,$$
$$\bar{u}^\circ(\mathring{p}) = u^\circ(\mathring{p}) - \rho^\circ(\mathring{e}') + \rho^\circ(\mathring{e}) - Q(\mathring{p}) \cdot T.$$

Note that the unchanged cyclic orders of the events allows us to use the Q values of the reference timetable. However, the planned event times $\rho^\circ + x$ do *not* necessarily fall in the interval $[0, T-1]$ any more.

Constraints (4) limit the running time extension of the timetable services between their terminal stations. Indeed, extended travel times are undesirable due to the longer travel times and due to the potentially higher rolling stock requirements. In (4), $\mathscr{S}$ denotes the set of services, while $\mathring{e}_s$ and $\mathring{e}'_s$ denote the departure and arrival events of service s at its terminal stations, respectively.

Finally, constraint (5) limits the cumulative running time extension. The choice of $M = 0$ corresponds to the requirement that the modified timetable's total running time supplement does not exceed that of the reference timetable.

2.3 Linear timetable

In this research, the robustness of the cyclic timetable is defined as the expected total delay experienced when it is operated subject to stochastic disturbances. In order to be able to describe our robustness measure, we need an evaluation framework for the planned timetable of a day.

We assume that the day's timetable consists of H repeated copies of the 1-h cyclic timetable. This results in the *linear timetable*. The set $\mathscr{E}$ of *linear events* is defined as

$$\mathscr{E} = \{(\mathring{e}, h) \mid \mathring{e} \in \mathscr{E}^\circ, 1 \leqslant h \leqslant H\}.$$

The set $\mathscr{E}_{\text{dep}}$ consists of the linear events associated with the departure of a train from a station, similarly, $\mathscr{E}_{\text{arr}}$ is the set of linear events associated with the arrival of a train. Throughout the paper, $\mathring{e}$ denotes the cyclic event corresponding to $e \in \mathscr{E}$.

The planned event times, subject to the event shift variables, are then given by a function $\pi : \mathscr{E} \to \mathbb{Z}$ defined as

$$\pi(\mathring{e}, h) = \rho^\circ(\mathring{e}) + x(\mathring{e}) + h \cdot T.$$

That is, π is an affine function of x. Note the time instants of a linear timetable lie on a linearly ordered time scale.

The set $\mathscr{P}$ of *linear processes* arises from the cyclic processes; their lower bounds form the only restrictions for operating the linear timetable. The set $\mathscr{P}$ is defined as

$$\begin{aligned}\mathscr{P} = &\left\{((\mathring{e}, h), (\mathring{e}', h)) \mid (\mathring{e}, \mathring{e}') \in \mathscr{P}^\circ, Q(\mathring{e}, \mathring{e}') = 0 \text{ and } 1 \leqslant h \leqslant H\right\} \\ &\cup \left\{((\mathring{e}, h), (\mathring{e}', h+1)) \mid (\mathring{e}, \mathring{e}') \in \mathscr{P}^\circ, Q(\mathring{e}, \mathring{e}') = 1 \text{ and } 1 \leqslant h \leqslant H\right\}.\end{aligned}$$

Indeed, the parameter Q determines whether or not a cyclic process crosses the boundary of the hour. We use the symbol $\mathring{p}$ to denote the cyclic process that naturally corresponds to $p \in \mathscr{P}$. The technically minimal process time is defined as $\ell(p) := \ell^\circ(\mathring{p})$ for each $p \in \mathscr{P}$.

The linear timetable can be seen as a directed graph on the vertex set $\mathscr{E}$ with the arc set $\mathscr{P}$. This graph is acyclic since each arc points from an earlier planned time instant to a later one. We assume that the set $\mathscr{E}$ is sorted by planned event time. Further, the graph representation immediately gives rise to the notions of outgoing and incoming processes of an event.

2.4 Delays and robustness

The time needed to carry out the linear processing has a stochastic character. Train drivers may choose different speed profiles, the disembarking and embarking at a station may vary depending on the number of passengers, and so on. We call such effects *disturbances* of a process; these are the primary causes of deviations from the planned timetable. A disturbance is a stochastic variable that increases the technically minimal process time. We assume that the disturbances have a known probability distribution and that they are independent.

Delays arise when the disturbance of a process exceeds its time supplement. In such a case, the delay may propagate along the outgoing processes. It is important to note that the delays of the events are measurable in practice, and are heavily correlated. Disturbances, on the other hand, cannot be observed immediately, they can be estimated from the collected delay data. The discussion of such estimation methods lies out of the scope of this paper. Our implementation relies on the probability distributions obtained from NS.

The delay penalty of an event, where $y(e)$ is its realised event times, is defined by

$$P(e) := \alpha \cdot \max\{0; y(e) - \pi(e)\} + \beta \cdot \max\{0; y(e) - (\pi(e) + \gamma)\}. \tag{6}$$

The delay penalty is a piece-wise linear convex function of the realised event time with non-negative parameters α, β and γ. A penalty of α is accounted for each delay minute, and an additional penalty of β is accounted for each delay above γ minutes. Our main motivation is the wish of NS to improve on the 3-min punctuality of the timetable. Therefore we will set $\gamma = 3$.

Suppose first that the disturbance vector $\delta : \mathscr{P} \to \mathbb{R}_+$ is known. Then the realised event times and the corresponding total delay penalty are computed by solving the following optimisation model.

$$D := \min \quad \sum_{e \in \mathscr{E}_{\text{arr}}} P(e) \tag{7}$$

$$\text{s.t.} \quad y(e') \geqslant y(e) + \ell(p) + \delta(p) \qquad \text{for each } p = (e, e') \in \mathscr{P} \tag{8}$$

$$y(e) \geqslant \pi(e) \qquad \text{for each } e \in \mathscr{E}_{\text{dep}} \tag{9}$$

$$y \in \mathbb{R}^{\mathscr{E}} \tag{10}$$

Constraints (8) arise from the disturbed technically minimal process times. Constraints (9) make sure that no train departs earlier than its announced departure time.

The total delay penalty D is a function of the event shift variables x and the disturbances δ. In robust timetabling, we aim at minimising the expected total delay penalty. However, this objective cannot be computed analytically in large problem instances. Therefore we follow the methodology of Kroon et al. (2008) in that the expectation $\mathbb{E}D(x, \delta)$ is approximated by a sample average:

$$\mathbb{E}D(x, \delta) \approx \frac{1}{N} \sum_{i=1}^{N} D(x, \delta_i)$$

where $\delta_1, \ldots, \delta_N$ are independently drawn random disturbance vectors.

Letting denote $\Phi(x) := \frac{1}{N} \sum_{i=1}^{N} D(x, \delta_i)$, the robust timetabling problem is a two-stage stochastic integer program defined as follows.

$$\min \quad \Phi(x) \tag{11}$$

$$\text{s.t.} \quad x \in \mathbb{Z}^{\mathscr{E}^\circ} \text{ satisfies } (2)-(5). \tag{12}$$

Notice that $\Phi(x)$ is a convex function of the decision variables x. Indeed, it is easy to verify that, for any disturbance vector δ, the optimum value of the convex optimisation model (7)–(10) is a convex function of x.

The substitution of the expectation by the sample average, known as the Sample Average Approximation Method, have been extensively studied. Detailed results on its soundness and its convergence properties can be found, for example, in Shapiro and Homem-De-Mello (2000) and Kleywegt et al. (2001).

2.5 Simulation algorithm

An efficient *simulation* algorithm is available for evaluating $\Phi(x)$ for any given vector x. An optimal solution of (7)–(10), subject to a disturbance vector δ_i, arises from the dynamic programming scheme of Algorithm 1. The algorithm makes sure that every event is carried out as soon as its predecessors allow it to happen. The only external restriction is that departures cannot take place earlier than their scheduled event time. The correctness follows from the acyclicity of the graph and from the monotone non-decreasing character of the objective function.

Algorithm 1: Simulation by dynamic programming (for given π and δ_i)

```
1 foreach e' ∈ ℰ do
2     if e' ∈ ℰ_dep then
3         Let y(e') := max{π(e'); max{y(e) + ℓ(p) + δ_i(p) | p = (e, e') ∈ 𝒫}}
4     else
5         Let y(e') := max{y(e) + ℓ(p) + δ_i(p) | p = (e, e') ∈ 𝒫}
```

Then the value $\Phi(x)$ is obtained by solving the dynamic program for N disturbance vectors. The time complexity of the simulation is thus $O(N \cdot |\mathscr{P}|)$.

3 Solution method

In this paper we propose a Branch-and-Bound algorithm for solving the robust cyclic timetabling problem. Our approach implements the common elements of Branch-and-Bound: we define nodes to represent subproblems of the overall optimisation problem, we compute lower bounds and feasible heuristic solutions at the nodes, and we keep replacing the nodes by new ones with smaller solution spaces.

The main structure of the algorithm is sketched in Algorithm 2. Each *node* v of the Branch-and-Bound tree is characterised by the restriction vectors $\alpha_v : \mathscr{E}^\circ \to \mathbb{Z}$ and $\omega_v : \mathscr{E}^\circ \to \mathbb{Z}$. The restriction interval $[\alpha_v(\mathring{e}); \omega_v(\mathring{e})]$ indicates the possible values of the integral-valued variable $x(\mathring{e})$ at that node. The initial node of the search has $\alpha_v(\mathring{e}) := -m(\mathring{e})$ and $\omega_v(\mathring{e}) := m(\mathring{e})$ for each $\mathring{e} \in \mathscr{E}^\circ$ (see line 1).

When branching on a node v, we select an event $\mathring{e}$, and we replace node v by a set of child nodes. The children have all possible single-point restriction intervals for event $\mathring{e}$, while they inherit all other restriction intervals from v (lines 16–22).

The best solution found so far, together with the nodes' lower bounds, is used to prune nodes that have suboptimal solutions only (lines 6, 27 and 29). The progress of the search is monitored by maintaining the best available global lower bound LB^* (line 9).

In what follows, we discuss the remaining major steps of the approach in details, and we propose a particular rule for node selection, variable selection as well as for a constructive heuristic. In Sect. 4 we compare the proposed rules to natural alternative rules.

Algorithm 2: High-level description of the Branch-and-Bound algorithm

```
1  Let s be the initial node with [α_s(ê); ω_s(ê)] := [−m(ê); m(ê)] for each ê ∈ ℰ°
2  Compute lower bound LB(s) for node s
3  Compute a feasible solution at node s with objective value UB(s)
4  if s is proven to be infeasible then
5      Stop
6  Let UB* := UB(s)
7  Let 𝒩 := {s}
8  while 𝒩 ≠ ∅ do
9      Let LB* := min{LB(v) | v ∈ 𝒩}
10     Select a node n ∈ 𝒩
11     Remove n from 𝒩
12     if n is not proven to be infeasible then
13         if There exists an event ê with α_n(ê) < ω_n(ê) then
14             Select an event ê with α_n(ê) < ω_n(ê)
15             Let 𝒞 := ∅
16             foreach j ∈ {α_n(ê), α_n(ê)+1, ..., ω_n(ê)} do
17                 Let v be a new, uninitialised node
18                 for f̊ ∈ ℰ° do
19                     if ê = f̊ then
20                         Let [α_v(f̊); ω_v(f̊)] := [j, j]
21                     else
22                         Let [α_v(f̊); ω_v(f̊)] := [α_u(f̊); ω_u(f̊)]
23                 Compute lower bound LB(v) for node v
24                 Compute a feasible solution at node v with objective value UB(v)
25                 if v is not proven to be infeasible then
26                     Add v to 𝒞
27                     Let UB* := min{UB*; UB(v)}
28             Add each v ∈ 𝒞 to 𝒩
29             Remove all nodes u from 𝒩 with LB(u) ⩾ UB*
```

3.1 Computing the nodes' lower bounds

In lines 2 and 23 of Algorithm 2, we consider node v of the Branch-and-Bound tree, and we compute a lower bound valid for every feasible solution x with $\alpha_v \leqslant x \leqslant \omega_v$.

The lower bound arises from two minor adjustments of the Algorithm 1. First, we pretend that $x(\mathring{e}) = \alpha_v(\mathring{e})$ holds for each departure event e. That is, the planned cyclic timetable has the weakest possible impact on the realisations. Second, we estimate the delay penalty of an arrival event e under the very conservative assumption $x(\mathring{e}) = \omega_v(\mathring{e})$. That is, delay penalties are counted only if the restriction intervals of node v trivially imply them.

The lower bound can be computed very quickly, in a matter of milliseconds, for large real-world problem instances. However, the lower bounds are rather weak.

They do not appear to help the Branch-and-Bound algorithm to close the optimality gap in reasonable time, not even after excessive branching.

In spite of their unsuitability for closing the gap, the lower bounds do play a crucial role in the node selection step (line 10) and in the variable selection step (line 14), as pointed out on the next sections.

3.2 Node selection

In line 10 of Algorithm 2, we select the next node to branch on. The scope of this paper is restricted to two elementary rules: depth-first-search (DFS) and best-bound-search (BBS). It is a subject of further research to study the plethora of advanced node selection rules, including problem-specific (or even instance-specific) ones.

The BBS rule selects a node with the lowest lower bound. That is, it chooses

$$\operatorname{argmax} \{LB(v) \mid v \in \mathcal{N}\}.$$

Thus the BBS rule attempts to push the global lower bound LB^* as quickly as possible.

The DFS rule select a node that was added as last to $\mathcal{N}$. That is, the search dives in the Branch-and-Bound tree greedily, and backtracks on fully explored nodes only. It must be noted that, due to the depth of the tree, the DFS rule has little chance to improve the global lower bound LB^* of real-world instances.

A critical detail of the DFS rule is to properly specify the order of the child nodes in which they are added to the tree (line 28). Indeed, the DFS search will continue on the last added child node.

In this paper we propose DFS in combination with the rule that the child nodes are ordered by decreasing lower bound $LB(v)$. That is, DFS continues on the child with the smallest lower bound. We hypothesise that the lower bounds capture the *local* consequences of the branching well, even though they may not be adequate as *global* lower bounds. Under this hypothesis, the best feasible solutions are likely to be found under the child node with the smallest lower bound.

3.3 Variable selection

In line 14 of Algorithm 2, we select the variable on whose possible values to branch on. We propose the following criterion for variable selection.

When node n is considered for branching, we select the event $\mathring{e}$ whose realised time in the lower bound simulation has the largest deviation from its lower restriction $\alpha_n(\mathring{e})$. More precisely, we define for each event $\mathring{e} \in \mathscr{E}^\circ$

$$\Delta(\mathring{e}) := \frac{1}{H}\sum_{h=1}^{H}\Big(y(\mathring{e},h) - (\rho^\circ(\mathring{e}) + \alpha_n(\mathring{e}) + h \cdot T)\Big).$$

Mind that $\rho^\circ(\mathring{e}) + \alpha_n(\mathring{e}) + h \cdot T$ is a natural comparison point for the realised time $y(\mathring{e},h)$ of linear event $(\mathring{e},h)$. In fact, we have $\rho^\circ(\mathring{e}) + \alpha_n(\mathring{e}) + h \cdot T \leqslant y(\mathring{e},h)$ if $(\mathring{e},h)$ is a departure.

Then, our variable selection rule picks

$$\operatorname{argmax}\{\Delta(\mathring{e}) \mid \mathring{e} \in \mathscr{E}^\circ \text{ with } \alpha_n(\mathring{e}) < \omega_n(\mathring{e})\}.$$

Indeed, the simulation algorithm is very likely to underestimate the contribution of this event to the total delay penalty. The branching takes place at a variable that does matter for the lower bounds. We expect that the increased accuracy of the lower bounds is beneficial for the subsequent node selection and variable selection steps.

3.4 Finding feasible solutions

In lines 3 and 24 of Algorithm 2, we construct a feasible solution by solving an auxiliary optimisation model. The objective of this model is designed to prefer solutions that are likely to have a low Φ value.

The auxiliary model aims at minimising the Euclidean distance from a target point z subject constraints (2)–(5) and subject to the restriction intervals $[\alpha_n; \omega_n]$. Formally, the auxiliary model reads as follows.

$$\begin{aligned}
&\text{minimise } \textstyle\sum_{\mathring{e}\in\mathscr{E}^\circ} (x(\mathring{e}) - z(\mathring{e}))^2 \\
&\quad\text{subject to } \alpha_n(\mathring{e}) \leqslant x(\mathring{e}) && \leqslant \omega_n(\mathring{e}) && \text{for each } \mathring{e} \in \mathscr{E}^\circ \\
&\qquad -m(\mathring{e}) \leqslant\ x(\mathring{e}) && \leqslant m(\mathring{e}) && \text{for each } \mathring{e} \in \mathscr{E}^\circ \\
&\qquad \vec{\ell}^\circ(\mathring{p}) \leqslant\ x(\mathring{e}') - x(\mathring{e}) && \leqslant \vec{u}^\circ(\mathring{p}) && \text{for each } \mathring{p} = (\mathring{e}, \mathring{e}') \in \mathscr{P}^\circ \\
&\qquad\quad x(\mathring{e}'_s) - x(\mathring{e}_s) && \leqslant M_s && \text{for each } s \in \mathscr{S} \\
&\qquad\quad \textstyle\sum_{s\in\mathscr{S}} \big(x(\mathring{e}'_s) - x(\mathring{e}_s)\big) && \leqslant M \\
&\qquad\qquad x : \mathscr{E}^\circ \to \mathbb{Z}
\end{aligned}$$

We linearise the quadratic model under the assumption $-2 \leqslant x \leqslant 2$. We introduce a new decision variable $w(\mathring{e})$ for each cyclic event $\mathring{e} \in \mathscr{E}^\circ$, and we replace the quadratic objective term $x(\mathring{e})^2$ by $w(\mathring{e})$. Further, we add the following constraints.

$$\begin{aligned}
w(\mathring{e}) &\geqslant x(\mathring{e}) \\
w(\mathring{e}) &\geqslant -x(\mathring{e}) \\
w(\mathring{e}) &\geqslant 3x(\mathring{e}) - 2 \\
w(\mathring{e}) &\geqslant -3x(\mathring{e}) - 2
\end{aligned}$$

These constraints form the convex hull of the points $\{(-2;4), (-1;1), (0;0), (1;1), (2,4)\}$ in the $x(\mathring{e}) - w(\mathring{e})$-space, omitting the constraint $w(\mathring{e}) \leqslant 4$ only. Therefore every optimal solution of the linearised model satisfies $w(\mathring{e}) = x(\mathring{e})^2$ for each $\mathring{e} \in \mathscr{E}^\circ$.

The components $z(\mathring{e})$ of the target point are defined as convex combinations of the values $\alpha_n(\mathring{e})$ and $\omega_n(\mathring{e})$:

$$z(\mathring{e}) = (1 - \lambda_{\mathring{e}})\alpha_n(\mathring{e}) + \lambda_{\mathring{e}}\omega_n(\mathring{e}) \qquad \text{for each } \mathring{e} \in \mathscr{E}^\circ.$$

The choice of the weights $\lambda_{\mathring{e}}$ has a major impact on the performance of our algorithms. In this paper we propose the weight structure of

$$\lambda_{\mathring{e}} = \begin{cases} 1 & \text{if } \mathring{e} \text{ is an arrival event,} \\ 0 & \text{if } \mathring{e} \text{ is a departure event.} \end{cases}$$

Our weight structure prefers solutions where departure events are scheduled as early as possible, and arrival events are scheduled as late as possible. This preference leads to the least delays if we neglect the interaction between different timetable services.

3.5 Constraint propagation

The performance of the Branch-and-Bound algorithm can be enhanced by letting the restriction intervals $[\alpha_n; \omega_n]$ propagate along the planning constraints (3). Suppose that we have a constraint $x(\mathring{e}') - x(\mathring{e}) \geqslant \ell$ and the restrictions

$$\alpha_n(\mathring{e}) \leqslant x(\mathring{e}) \leqslant \omega_n(\mathring{e}),$$
$$\alpha_n(\mathring{e}') \leqslant x(\mathring{e}') \leqslant \omega_n(\mathring{e}').$$

Then we can replace the restriction intervals by tighter intervals

$$\alpha_n(\mathring{e}) \leqslant x(\mathring{e}) \leqslant \min\{\omega_n(\mathring{e}); \ell + \omega_n(\mathring{e}')\},$$
$$\max\{\alpha_n(\mathring{e}'); \ell + \alpha_n(\mathring{e})\} \leqslant x(\mathring{e}') \leqslant \omega_n(\mathring{e}').$$

We apply this tightening step at each node for each planning constraint as long as any improvement is possible. This prevents the algorithm to branch into infeasible nodes, and helps computing tighter lower bounds. The time requirement of the propagation steps is a negligible fraction of the overall computation time.

4 Computational results

In this section we illustrate the behaviour of our Branch-and-Bound algorithm on large problem instances of NS. For the sake of clarity, we report here computational results on a single test instance. We present further results in the Appendix; the additional results are fully in line with the content of this section.

The test instance concerns the 1-h timetable of the entire Dutch rail network. The timetable contains about 10,000 departure events and the same number of arrival events, connected by 29,000 processes. All input data, including the probability distribution of the disturbance, is obtained from SOM, NS's commercial implementation of the model of Kroon et al. (2008). We evaluate the robustness of the timetable averaging 120 independent replications of a day, where a simulated day consists of 20 h. The linear timetable has about 180,000 events and 420,000 processes, 170,000 of which have a disturbance.

We implemented the Branch-and-Bound algorithm in the Java programming language. The computations were performed on a standard PC, equipped with an Intel Core i7-2600 3.4GHz processor and with 16 GB of memory. The programme code uses 7 threads when evaluating the objective values and the lower bounds; it is a single-threaded implementation in every other aspect. The running times reported

below are elapsed real times: they measure the time difference between starting the algorithm and receiving the solution.

Note that SOM cannot handle more than 120 replications of a day before running out of its memory of 100 GB, and the computation time amounts to several days. In contrast, the memory consumption of our algorithm does not depend on the number of replications, while the computation time increases linearly.

Unfortunately, it was not possible to perform a direct comparison between SOM and our Branch-and-Bound algorithm, mainly due to the difficulty to access the random disturbances in a SOM run. A more detailed evaluation of our algorithm is the subject of our on-going research.

4.1 Overview of the best solution

Table 1 compares the reference solution to the best solution found. We use here all settings proposed in Sect. 3. Our algorithm was able to improve on the total delay penalty; more than 60% of the absolute gap disappears.

Figure 1 illustrates the improvement of the objective value of the best solution in course of the Branch-and-Bound algorithm. The horizontal axis corresponds to the number of visited nodes (below) and to the elapsed time in minutes (above). The vertical axis is the objective value itself. Note that, due to the rather greedy nature of

Table 1 Comparison between the reference solution and the best solution we found

	Reference solution	Best solution
Total delay penalty (sample mean)	7362	6290
Confidence interval for 95%	(6779; 7946)	(5753; 6827)
Proven lower bound	5632	5657
Absolute optimality gap	1730	633
Relative optimality gap	23.5%	10.1%

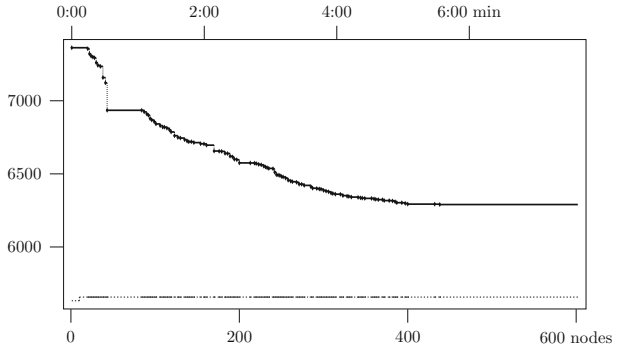

Fig. 1 Progression of the incumbent objective value and that of the lower bound. The axes indicates the number of visited Branch-and-Bound nodes (*below*) and the elapsed time (*above*) as well as the total delay penalty (*left*)

the DFS rule, no further improvement could be found, even when letting the branching continue for another 4000 nodes.

The figure also indicates that lower bound of the initial node is barely improved upon, and that a considerable optimality gap remains. Tests on other instances show gaps as high as 30%. We observed that large relative gaps tend to occur in cases when the reference solution shows very little dispunctuality due to unrealistically low external disturbances.

Table 1 shows that the 95%-level confidence intervals of the objective value are rather wide; the width is more than 15% of the mean value. This indicates that the sample size of 120 may be insufficient. While an analysis on the reliability of the results lies out of our scope, we want to mention that the Branch-and-Bound algorithm is very well suitable for sample sizes much larger than 120. In fact, the computation times scale linearly with the sample size.

4.2 Computation times

Our Branch-and-Bound algorithm finds the rather significant improvement of the reference solution quite quickly. The best solution is reached after 5 min, and slightly inferior solutions are available even earlier. This is in line with our goal of developing a pragmatic approach for robust timetabling.

In course of the algorithm, a node is evaluated in 0.63 s on average (with very little variance). Finding a feasible solution requires 0.20 s: we call the commercial MIP solver CPLEX on the model in Sect. 3.4. The linearised quadratic integer programs in our application have a few thousand variables and a relatively simple structure; modern MIP solvers can solve them quickly and reliably.

An additional 0.41 s per node are spent on the simulation algorithm to evaluate the feasible solution and to compute the lower bound. Mind that 120 replications of the 20-h days are evaluated in 0.41 s. We achieve this running times by having a multi-threaded implementation, by employing fast data structures.

4.3 Node selection and variable selection rules

In Fig. 2 we compare the node selection rules BBS and DFS, assuming that all other settings are identical to those used in Sect. 4.1. In spite of a few initial lucky steps, the BBS rule guides the search in a fruitless direction. The BBS rule does help finding better lower bounds but the difference to the DFS bounds is small (see the dotted lines on the bottom of Fig. 2).

Next, we compare the proposed variable selection rule to some natural alternatives.

- Rule R: The rule we propose in Sect. 3.3.
- Rule G: Select a variable with the largest gap between its upper and lower restriction (i.e., maximise $\omega_n(\mathring{e}) - \alpha_n(\mathring{e})$). This rule is motivated by the wish to create tightly constrained child nodes.
- Rule A: Give preference to arrival events; otherwise fall back to Rule G. This rule is motivated by the observation that arrival events play a crucial role in the evaluation of the objective value.

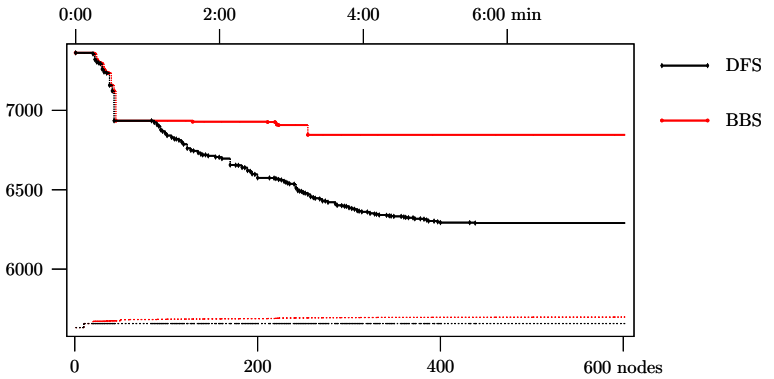

Fig. 2 Progression of the objective value when using BBS and DFS as node selection rule

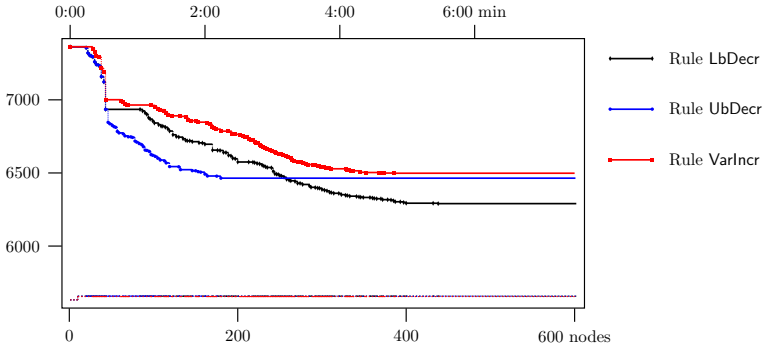

Fig. 3 Progression of the objective value when using LbDecr, UbDecr and VarIncr as child sorting rule

It turns out that Rule R clearly outperforms the other candidates. The same observation holds for our other test instances, as well (see Appendix).

As mentioned in Sect. 3.2, the DFS rule requires a carefully selected order in which to process the child nodes after branching. We compare the proposed child sorting rule to two natural candidates.

- Rule LbDecr: Sort the child nodes by decreasing lower bound; this is the rule we propose in Sect. 3.3.
- Rule UbDecr: Sort the child nodes by decreasing upper bound where the upper bound is the objective value of the heuristic solution found at that node. This rule drives the search to a child that has proved to have a better heuristic solution.
- Rule VarIncr: Sort the child nodes by increasing value for the variable to branch on. We do not have any clear intuition for this rule.

Figure 3 depicts the progression of solutions according to these three rules. The rules have a comparable performance, there is no clear winner, although Rule LbDecr does tend to have the slight edge in our other instances, as well.

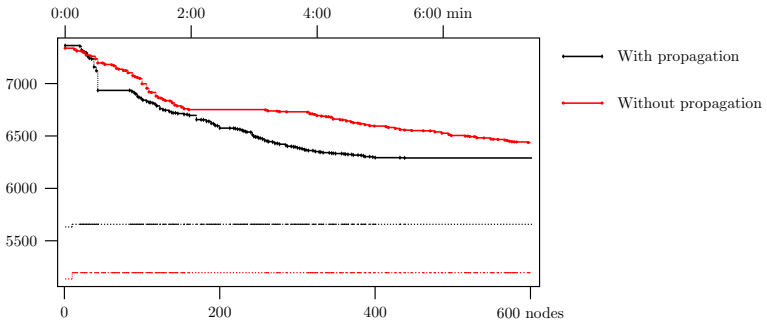

Fig. 4 Progression of the objective value with and without using constraint propagation. Note that the better lower bound belongs to the case with constraint propagation (Colour figure online)

We want to point out that not all sorting rules perform well. In particular, hardly any improvement on the reference solution is achieved when applying the opposite of the three compared rules (i.e., by increasing lower bound, by increasing upper bound and by decreasing variable value).

The results for the variable selection rule and for the child sorting rule indicate that the lower bounds can indeed be exploited in order to guide the Branch-and-Bound search towards better solutions. This benefit fully justifies the effort to compute the bounds, even if they do not lead to tight optimality gaps.

4.4 Constraint propagation

As explained in Sect. 3.5, we propose a simple constraint propagation method to tighten the lower and upper restrictions at each node. In Fig. 4 we compare the outcome of our algorithm to a variant where constraint propagation is not performed.

The most striking difference lies in the lower bounds: constraint propagation leads to a lower bound of 5631 at the initial node, the lower bound without using constraint propagation is 5135. We also observe that, by applying constraint propagation, the search is prevented from going to infeasible direction, and this indeed shows in the progression of the incumbent objective value.

5 Conclusions and future research

In this paper we propose a Branch-and-Bound algorithm for solving the stochastic programming model of Kroon et al. (2008) for robust cyclic timetabling. Our aim is to find good solution for real-life problem instances of Netherlands Railways (NS). Note that these instances are barely tractable by the Mixed Integer Programming approach of Kroon et al. (2008) due to the excessive computation times and memory consumption.

The various components of the Branch-and-Bound algorithm (such as the node selection rule, the variable selection rule and the constructive heuristics) are specific

to the instances of NS. We do expect, though, that the main ideas carry over to other application areas, too.

Our approach is an exact algorithm although it is unlikely to prove the optimality of the found solutions (or even to find optimal solutions) because of the weak lower bounds we compute at the nodes of the Branch-and-Bound tree. Therefore, our approach is best described as a heuristic algorithm.

Computational tests indicate that our algorithm is able to handle the large problem instances of NS. We improve the robustness of the reference solution by a large amount in a matter of a few minutes. The proposed algorithm has great potential to become a valuable addition to the timetable planners' tool-set due to its short computation time and due to its predictable behaviour.

Our on-going research focuses on thorough computational tests to demonstrate the true practical impact of the Branch-and-Bound algorithm, as well as on extending its capabilities. In particular, we are interested in adjusting the algorithm for a larger solution space where the events can be shifted by more than 1 min. Moreover, we are also working on an alternative algorithmic approach: We formulate the optimisation problem as a convex integer program which is solved by the subgradient projection method combined with a rounding heuristic.

Acknowledgements The author thanks Pieter-Jan Fioole and Joël van't Wout of Netherlands Railways for the numerous discussions and for providing the input data of the computational tests. The author also wants to thank the late Gábor Maróti, Sr. for his suggestions concerning this paper.

Appendix: Further computational results

In this section we summarise our results on a larger set of instances. We consider two large test cases (L1 and L2), each of them referring to a timetable for the entire Netherlands. Further, we consider three small test cases (S1, S2, S3), each of them referring to the timetable for a part of the Netherlands. The cases S1, S2, S3 cover three different geographical regions, each of them amounts to approximately one-eights of the country.

Within each of the five test cases, we carry out experiments with three scenarios: small disturbances, medium disturbances and large disturbances. We indicate these scenarios by suffixes A, B and C. The medium scenarios' mean disturbances are equal to those in the commercial implementation SOM; the small and large scenarios feature lighter and heavier disturbances, respectively. Altogether, we deal with 15 test instances. The instances L1-B and L2-B are the most relevant ones from a practical point of view. We note that Sect. 4 discusses the outcome for L1-B.

Table 2 Reference solution's objective value, best solution's objective value, proven lower bound and average delay penalty for test cases L1, L2, S1, S2 and S3; suffixes A, B and C indicate small disturbances, medium disturbances and large disturbances, respectively

Case	Lower bound	Reference			Best		
		Value	Gap (%)	Avg	Value	Gap (%)	Avg
L1-A	480	931	48.4	0.06	695	30.9	0.05
L1-B	5657	7363	23.2	0.49	6290	10.1	0.42
L1-C	32,632	37,218	12.3	2.46	34,353	5.0	2.47
L2-A	462	898	48.5	0.07	647	28.6	0.05
L2-B	4591	6315	27.3	0.47	5366	14.4	0.40
L2-C	22,688	26,641	14.8	1.98	24,957	9.1	1.85
S1-A	150	245	38.9	0.12	206	27.4	0.10
S1-B	402	569	29.5	0.27	488	17.8	0.23
S1-C	1493	1827	18.3	0.87	1649	9.5	0.79
S2-A	147	233	37.0	0.11	168	12.6	0.08
S2-B	514	710	27.6	0.33	548	6.2	0.26
S2-C	930	1190	21.8	0.56	972	4.3	0.46
S3-A	118	202	41.7	0.09	148	20.7	0.07
S3-B	372	551	32.5	0.25	436	14.6	0.20
S3-C	788	1095	28.0	0.50	889	11.4	0.41

Table 2 presents the reference solution's objective value, the best solution's objective value and the proven lower bound for each of the 15 test instances. In addition, the table gives the average delay penalty, defined as the objective value divided by the number of arrival events. Recall that the penalty structure gives 1 unit of penalty for each minute of delay, and 3 units of penalty for each minute of delay above 3 min.

We observe that the Branch-and-Bound algorithm consistently achieves a major improvement. Medium and large disturbance scenarios arrive at optimality gaps of 10–15%, with occasional outliers of 4–18%. However, instances with small disturbances show relative gaps as high as 30%. These are the instances where the reference timetable performs in absolute terms very well, the average delay of the trains amounts to a couple of seconds. The lower bounds are based on a rough relaxation of the problem; they do capture the larger delays but they struggle to see the smaller ones. We believe that the relative gaps of 25% or higher are mainly caused by the weak lower bounds.

Figures 5, 6 and 7 compare our choices for the node selection rules, variable selection rules and child sorting rules to natural alternatives; these figures are the counterparts of Figs. 2, 3 and 8. As before, the black lines indicate the proposed rules.

We observe in Fig. 5 that the DFS node selection rule heavily outperforms BBS on the large cases L1 and L2. The difference is less pronounced in the small cases S1, S2 and S3, but DFS still produces better solutions more quickly. However,

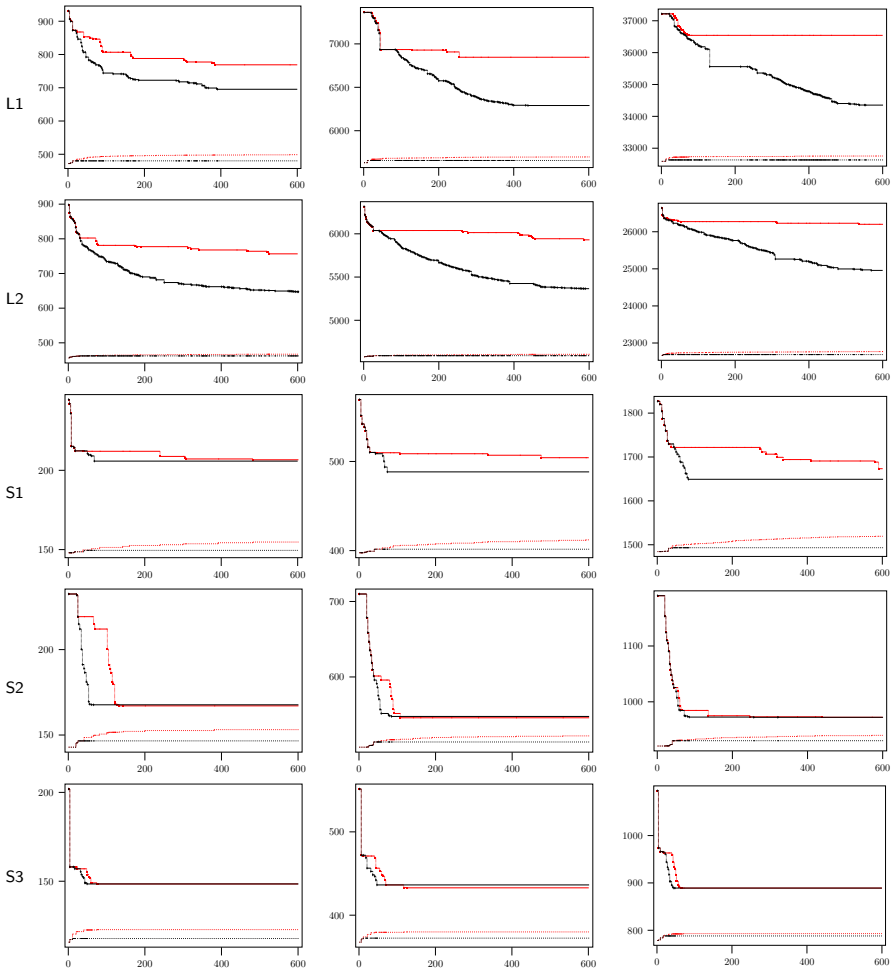

Fig. 5 Progression of the objective value when using BBS (*red line*) and DFS (*black line*) as node selection rule in test cases L1, L2, S1, S2 and S3. *Left column* small disturbances, middle column: medium disturbances, *right column* large disturbances (colour figure online)

DFS does provide noticeably sharper lower bounds in the small cases. The likely reason is that the small cases give rise to much shallower Branch-and-Bound trees than the large cases.

Figure 6 illustrates the superiority of the rule R in all test instances. Figure 7 shows a less conclusive picture for the child sorting rules. The proposed rule LbDecr is rarely worse than the other choices (and never *much* worse), but performs decisively better in 4 of the 15 instances.

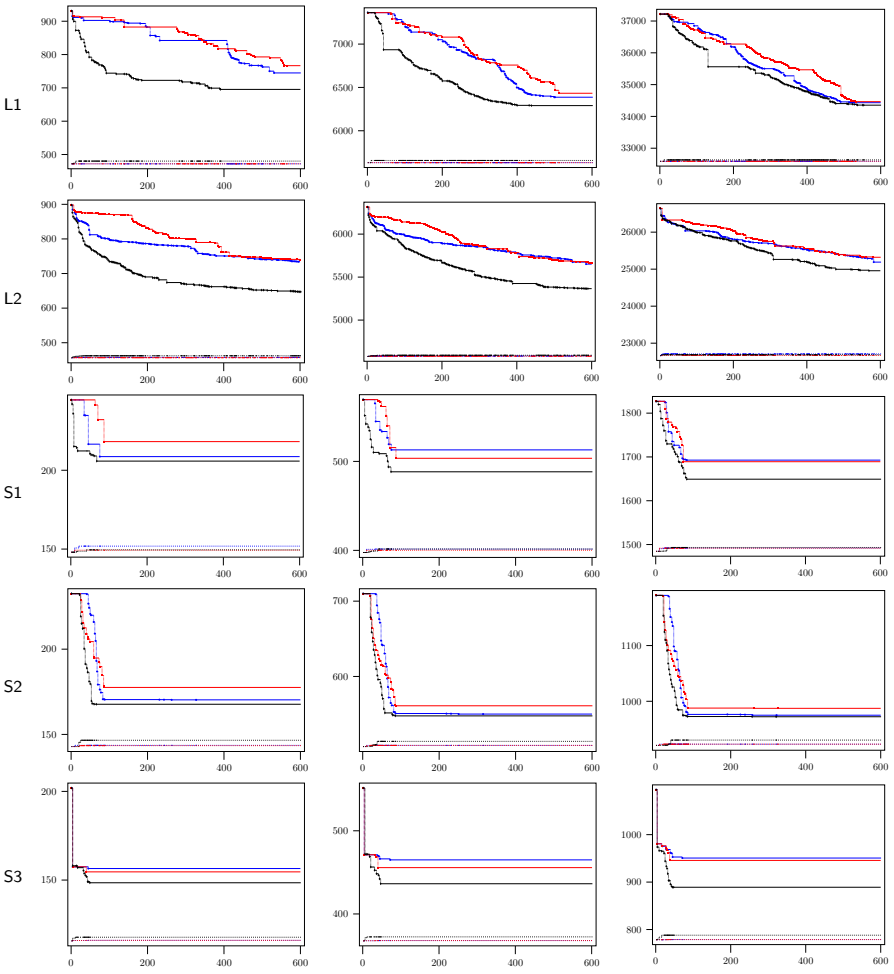

Fig. 6 Progression of the objective value when using rules R (*black line*), A (*blue line*) and G (*red line*) as variable selection rule in test cases L1, L2, S1, S2 and S3. *Left column* small disturbances, *middle column* medium disturbances, *right column* large disturbances (colour figure online)

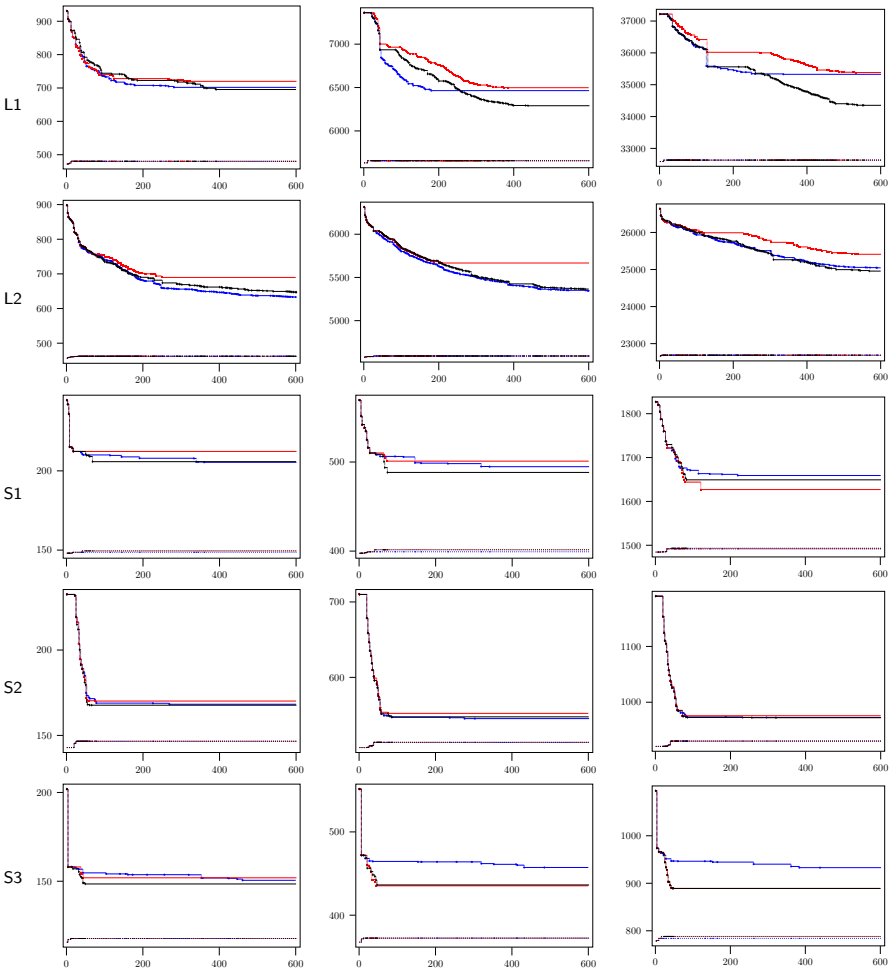

Fig. 7 Progression of the objective value when using LbDecr (*black line*), UbDecr (*blue line*) and VarIncr (*red line*) as child sorting rule in test cases L1, L2, S1, S2 and S3. *Left column* small disturbances, *middle column* medium disturbances, *right column* large disturbances (colour figure online)

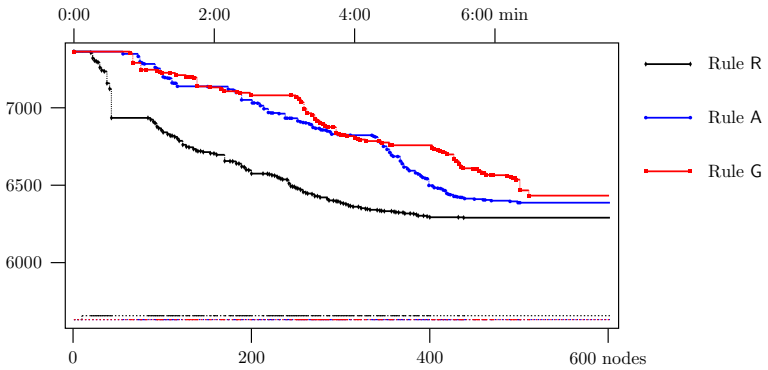

Fig. 8 Progression of the objective value when using rules R, A and G as variable selection rule

References

Cacchiani V, Toth P (2012) Nominal and robust timetabling problems. Eur J Oper Res 219:727–737

Caprara A, Kroon L, Monaci M, Peeters M, Toth P (2007) Passenger railway optimization. In: Barnhart C, Laporte G (eds) Handbook in OR and MS, vol 14. Elsevier, Amsterdam, pp 129–187

Cicerone S, D'Angelo G, Stefano GD, Frigioni D, Navarra A (2009) Recoverable robust timetabling for single delay: complexity and polynomial algorithms for special cases. J Combin Optim 18:229–257

Fischetti M, Salvagnin D, Zanette A (2009) Fast approaches to improve the robustness of a railway timetable. Transp Sci 43:321–335

Goerigk M (2014) Exact and heuristic approaches to the robust periodic event scheduling problem. Public Transp 7:101–119

Kleywegt A, Shapiro A, Homem-De-Mello T (2001) The sample average approximation method for stochastic discrete optimization. SIAM J Optim 12(2):479–502

Kroon L, Maróti G, Helmrich MR, Vromans M, Dekker R (2008) Stochastic improvement of cyclic railway timetables. Transp Res Part B 42(6):553–570

Kroon L, Huisman D, Maróti G (2014) Optimisation models for railway timetabling. In: Hansen I, Pachl J (eds) Railway timetable and operations, 2nd edn. DDV Media Group, Hamburg

Kümmling M, Großmann P, Nachtigall K, Opitz J, Weiß R (2015) A state-of-the-art realization of cyclic railway timetable computation. Public Transp 7(3):281–293

Liebchen C (2006) Periodic timetable optimization in public transport, Ph.D. thesis. Technische Universität Berlin

Liebchen C, Schachtebeck M, Schoebel A, Stiller S, Prigge A (2010) Computing delay resistant railway timetables. Comput Oper Res 37(5):857–868

Nachtigall K, Opitz J (2008) Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations. In: Fischetti M, Widmayer P (eds) 8th Workshop on algorithmic approaches for transportation modeling, optimization, and systems (ATMOS'08), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl. OpenAccess Series in Informatics (OASIcs), vol 9. http://drops.dagstuhl.de/opus/volltexte/2008/1588

Peeters L (2003) Cyclic railway timetable optimization, Ph.D. thesis. Erasmus University Rotterdam, Rotterdam School of Management, The Netherlands

Schrijver A, Steenbeek A (1994) Timetable construction for RailNed. Technical report, Center for Mathematics and Computer Science, Amsterdam, The Netherlands **(in Dutch)**

Serafini P, Ukovich W (1989) A mathematical model for periodic scheduling problems. SIAM J Discrete Math 2(4):550–581

Shafia M, Aghaee M, Sadjadi S, Jamili A (2012) Robust train timetabling problem: mathematical model and branch and bound algorithm. IEEE Trans Intell Transp Syst 13(1):307–317

Shapiro A, Homem-De-Mello T (2000) On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. SIAM J Optim 11(1):70–86

Vromans M (2005) Reliability of railway systems, Ph.D. thesis. Erasmus University, Rotterdam